python_grammar

Holy Frege

October 12, 2021

Contents

1 Python Grammar

grammar는 외워야 한다.

2 (1) Running Python

2.1 python interpreter를 실행시킨다.

2.2 print "hello world"

3 (2) Variables and Arithematic Expressions

3.1 Data Type

integer, float, string, boolean 을 BISF (B is F)를 data type이라고 하지만, python에선 data type을 기술하지 않는다. Dynamic typed language이기 때문이다. variable와 변수를 사용해서 표현할 줄 만 알면 된다. 아래 program에서 variable들의 type은 value에 의해 결정되는 것을 알아야 한다. 그리고 나중에 나오겠지만, while문에서 :은 block의 시작을 의미한다. 따라서 : 다음의 명령어는 indentation이 필요하다.

- 이자율 계산 program

```
principal = 1000
rate = 0.05
numyears = 5
year =1
while year <= numyears:
  principal = principal * (1 * rate)
  print(year,principal)
year +=1
```

4 (3) Data Structures

4.1 Strings (array에 해당)

- create

string에는 문자열을 ' ' 이나 " " 안에 넣으면 된다. 그런데 만일 문자열에 ' ' 이나 " " 이 포함되어 있다면? "' triple quote를 사용하면 된다. 단순하게 생각하자. quote가 포함된 문자열과 quote가 포함되지 않은 문자열만 있다고.

```
a= "Hello World"
b= " python is 'groovy'"
d = ''' 'python' is "good" '''
print(d)
```

- read read는 index를 사용해서 읽는다. index는 2가지가 있다. :을 사용하는 범위가 있는 index와 범위가 없는 index

  ```
  a = "This is a python"
  print(a[2])
  print(a[2:5])
  print(a[:8])
  print(a[3:])
  ```

- update update라고 하긴 그렇지만, + operator(concatenate)를 사용해서 문자열을 합칠 수 있다.

  ```
  a = "abc"
  b= "def"
  print (a + b)
  ```

- delete

delete라고 하긴 그렇지만, string이 아닌 다른 것으로 변환할 수 있다.

```
a = "30"
b= "35"
c = int(a) + int(b)
print(c)
```

4.2 Lists [ ]

- create

  ```
  names = ["Dave", "Mark", "Ann", "Phil"]
  print(names)
  ```

- read string(array)과 비슷하다. index를 사용한다. 범위가 있는 index, 범위가 없는 index처리가 array와 동일하다. 다만 범위가 있는 index의 경우 list를 return한다. 또한 unpack방식도 있다.

```
names = ["park", "kim", "lee"]
name1,name2,name3 =names
print(name1)
print(name2)
print(name3)
print(names[0])
print(names[0:1])
print(names[1:])
print(names[:2])
```

- update update는 3가지 방식이 있다. append(), insert(), index로 대입. index로 직접 추가시 문제가 있음. 범위가 있는 경우는 list형태로, 범위가 없는 경우는 값만을 삽입해야 한다. concatenate도 가능하다.

```
names = ["kim", "park", "lee"]
names.append("hwang")
print(names)
names.insert(2,"you")
print(names)
names[0:1] = ["oh","lim"]
print(names)
names = ["holy","jaemyung"] + ["sukyoel", "junpyo"]
print(names)
```

- delete 특별히 delete에 관한 것은 없다.
- example

```
import sys
if len(sys.argv) !=2:
     print("please supply a file name")
     raise SystemExit(1)
f= open(sys.argv[1])
lines = f.readlines()
f.close()

fvalues = [float(line) for line in lines]
```

```
print( "The minimum value is ", min(fvalues))
print("The maximum value is " , max(fvalues))
```

4.3 Tuples ()

- create

```
stock = ('good' , 100, 490.10)
temp = ('test',)
print(stock)
print(temp)
```

- read

tuple도 list의 indexing을 사용할 수 있다. 사용법도 동일하다. indexing이 범위가 있는 경우, 범위가 없는 경우에 따라서 값을 읽을수도 tuple을 읽을수도 있다. 또한 unpack방식도 사용할 수 있다.

```
names = ("holy", 12, 233.0)
name, age, weight = names  ;;unpack

print(names[1])
print(names[1:2])
print(name)
print(age)
print(weight)
```

- update
- delete

4.4 Dictionaries {}

4.5 Sets{}

- create set은 만들어질때, 원소를 packing해서 만들기 보단, 기존에 있던 collection을 인자로 넣어서 만든다. 특이한 점은 결과가 정렬이 된다는 것이다. 아래 결과는 좀 다르다.

```
r = set("hello")
s = set([30,2,20,20])
```

```
t = set((30,20,2,3))
print(r)
print(s)
print(t)
```

5 (4) File Input/Output

6 (5) Conditionals

python은 switch 나 case같으 muptiple testing을 할 수 없다. 여기서 알아야 하는건, if else문과 if elif else문이다.

- if else문

```
a = 30
b = 4

if a<b:
   pass
else:
   print(" computer says no!")
```

- if elif else

```
suffix = ".htm"
content = "abc"
if suffix == ".htm":
content = "text/html"
elif suffix == ".jpg":
content = "image/jpeg"
elif suffix == ".png":
content = "image/png"
else:
raise RuntimeError("Unknown content type")

print(content)
```